Rule 131 Declaration of Charles E. Gotlieb

I, Charles E. Gotlieb, am a U.S. citizen over 21 years of age, do declare:

I am the attorney of record for U.S. Patent Application serial number 09/578,672, and represent the assignee of that application.

I took a disclosure from the applicants of U.S. Patent Application serial number 09/578,672 and produced at least a draft of said patent application prior to December 30, 1999. A redacted copy of said draft is attached as Exhibit A. The redacted copy does not have new material. The draft shows all of the claimed elements of all of the claims as indicated below.

The current claims are reproduced below. Next to each current claim element in parenthesis is one location in the draft of Appendix A where that claimed element is shown. There may be other locations in that draft where claimed elements are shown.

Therefore, the applicants had in their possession the invention described in all claims prior to December 30, 1999, which is prior to the filing date of the Vick referemce U.S. 7,082,532, filed December 30, 1999.

All statements made herein of my own knowledge are true, and all statements made herein on information and belief are believed to be true.

I hereby declare that the above statements were made with the knowledge that willful false statements and the like are punishable by fine, imprisonment, or both (18 U.S.C. 1001) and may jeopardize the validity of application serial number 08/884,107 or any patent issuing thereon.

1. A method of processing a first request for web page, comprising:

receiving the first request for the web page(page 4, lines 16-17); and

transmitting, to a device from which the first request was received, at least one command to send a second request for the web page, and a first timestamp (page 4, lines 17-19).

2. The method of claim 1 wherein the transmitting step is responsive to an existence of a second timestamp received with the request (page 15, lines 21-22; page 16, lines 18-20; page 19, lines 7-9).

3. The method of claim 2 comprising the additional steps of:

identifying a third timestamp (page 19, lines 13-14); and

responsive to the second timestamp received with the request, processing the request for the web page responsive to the second timestamp and the third timestamp (page 19, lines 14-18).

4. The method of claim 3 wherein the identifying the third timestamp step is responsive to a capacity of at least one selected from at least one server and a device coupled to the at least one server (page 21, line 17-page 33, line 16).

5. The method of claim 4 additionally comprising incrementing at least one of a plurality of counters responsive to the first request (page 24, line 17-21).

6. The method of claim 5 wherein each of the plurality of counters corresponds to a range of time different from the other plurality of counters (page 24, lines 17-21).

7. The method of claim 6 wherein the identifying the third timestamp step is additionally responsive to at least one of the plurality of counters (page 30, line 21-page 33, line 15).

8. The method of claim 5 comprising the additional steps of:

receiving a notification of abandonment of at least one selected from the first request and the second request (page 25, lines 18-22); and

decrementing at least one of the plurality of counters (page 25, lines 18-22; page 28, lines 6-13; page 34, lines 4-14).

9. The method of claim 3 wherein the identifying the third timestamp step comprises sending a command to at least one selected from at least one server and a device coupled to the at least one server (page 29, lines 1-3; page 22, lines 1-6).

10. The method of claim 3 wherein the identifying the third timestamp step comprises building a file comprising a status of at least one selected from at least one server and at least one device coupled to the at least one server (page 29, lines 1-3; page 22, lines 1-6).

11. The method of claim 1, wherein the transmitting step is responsive to a type of the first request (page 15, lines 1-20).

12. The method of claim 1, additionally comprising transmitting computer readable program code devices

configured to cause a computer to send the second request responsive to the indicator transmitted (page 17, line 2 - page 19, line 6).

13. The method of claim 1 wherein the computer readable program code devices configured to cause the computer to send the second request responsive to the indicator transmitted comprise at least one selected from a Javascript script and a Java applet (page 17, line 2 - page 19, line 6).

14. A computer program product comprising a computer useable medium having computer readable program code embodied therein for processing a first request for web page, the computer program product comprising (page 7, lines 13-14; page 8, lines 9-20):

computer readable program code devices configured to cause a computer to receive the first request for the web page (page 4, lines 16-17); and

computer readable program code devices configured to cause a computer to transmit, to a device from which the first request was received, at least one command to send a second request for the web page, and a first timestamp (page 4, lines 17-19).

15. The computer program product of claim 14 wherein the computer readable program code devices configured to cause a computer to transmit are responsive to an existence of a second timestamp received with the request(page 15, lines 21-22; page 16, lines 18-20; page 19, lines 7-9).

16. The computer program product of claim 15 additionally comprising computer readable program code devices configured to cause a computer to:

identify a third timestamp(page 19, lines 13-14); and

responsive to the second timestamp received with the request, process the request for the web page responsive to the third timestamp and the second timestamp (page 19, lines 14-18).

17. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp are responsive to a capacity of at least one selected from at least one server and a device coupled to the at least one server (page 21, line 17-page 33, line 16).

18. The computer program product of claim 17 additionally comprising computer readable program code devices configured to cause a computer to increment at

least one of a plurality of counters responsive to the first request(page 24, line 17-21).

19. The computer program product of claim 18 wherein each of the plurality of counters corresponds to a range of time different from the other plurality of counters (page 24, lines 17-21).

20. The computer program product of claim 19 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp are additionally responsive to at least one of the plurality of counters (page 30, line 21-page 33, line 15).

21. The computer program product of claim 18 additionally comprising:

computer readable program code devices configured to cause a computer to receive a notification of abandonment of at least one selected from the first request and the second request (page 25, lines 18-22); and

computer readable program code devices configured to cause a computer to decrement at least one of the plurality of counters (page 25, lines 18-22; page 28, lines 6-13; page 34, lines 4-14).

22. The computer program product of claim 16 wherein the computer readable program code devices configured to

cause a computer to identify the third timestamp comprise sending a command to at least one selected from at least one server and a device coupled to the at least one server (page 29, lines 1-3; page 22, lines 1-6).

23. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp comprise computer readable program code devices configured to cause a computer to build a file comprising a status of at least one selected from at least one server and at least one device coupled to the at least one server (page 29, lines 1-3; page 22, lines 1-6).

24. The computer program product of claim 14, wherein the computer readable program code devices configured to cause a computer to transmit are responsive to a type of the first request (page 15, lines 1-20).

25. The computer program product of claim 14, additionally comprising computer readable program code devices configured to cause a first computer to transmit computer readable program code devices configured to cause second computer to send the second request responsive to the indicator transmitted (page 17, line 2 - page 19, line 6).

26. The computer program product of claim 14 wherein the computer readable program code devices configured to cause the computer to send the second request responsive to the indicator transmitted comprise at least one selected from a Javascript script and a Java applet (page 17, line 2 - page 19, line 6).

27. An apparatus for processing a first request for a web page, the apparatus comprising:

a user request router having an input coupled to an apparatus input operatively coupled for receiving the first request, the user request router for providing at an output a signal responsive to the first request received at the user request router input (page 15, line 1 - page 16, line 21); and

a cookie/applet generator having an input coupled to the user request router output for receiving the signal, the cookie/applet generator for providing, to a device from which the first request was received, via a first output coupled to an apparatus output, a first indicator of at least one time to send a second request for the web page (page 16, line 21 - page 19, line 6).

28. The apparatus of claim 27, wherein the first request comprises a second indicator of time, and the user

request router provides the signal at the user request router output responsive to the second indicator of time (page 15, lines 21-22; page 16, lines 18-20; page 19, lines 7-9).

29. The apparatus of claim 28, wherein the cookie/applet generator provides at a second output a third indicator of time corresponding to the first indicator of time (page 21, lines 3-7), the apparatus additionally comprising:

a strokecount storage for having an input coupled to the cookie/applet generator third output for receiving the third indicator of time, the strokecount storage for storing the third indicator of time and a set of fourth indicators of time and for providing the third indicator of time and the set of fourth indicators of time at an input/output (page 24, lines 7-12); and

a cutoff timestamp calculator having an input operatively coupled for receiving an indicator of capacity, the cutoff timestamp calculator for selecting and providing at an output a timestamp from the set of fourth indicators of time responsive to the capacity (page 31, line 7 - page 33, line 16); and

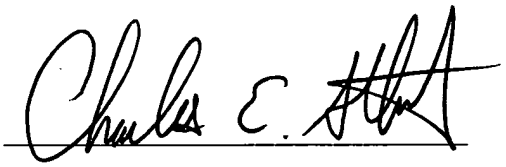
wherein the user request router additionally comprises a cutoff timestamp input coupled to the cutoff timestamp calculator output and the user request router provides the signal additionally responsive to the timestamp received at the cutoff timestamp input (page 19, line 7 - page 20, line 8).

30. The apparatus of claim 27, wherein the cookie/applet generator additionally provides at the cookie/applet generator first output computer readable program code devices configured to cause a computer to send the second request responsive to the indicator (page 17, line 2 - page 19, line 6).

31. The apparatus of claim 30 wherein the computer readable program code devices configured to cause the computer to send the second request responsive to the indicator transmitted comprise at least one selected from a Javascript script and a Java applet (page 17, line 2 - page 19, line 6).

Respectfully submitted,

August 28, 2009

By: 

Charles E. Gotlieb
Registration No. 38,164
Innovation Partners
540 University Ave., Suite 300
Palo Alto, CA 94301
(650) 328-0100

Appendix A

METHOD AND APPARATUS FOR CONTROLLING ACCESS TO A WEBSITE

Attorney Docket Number

1112

Express Mail Label Number

Inventors

Related Applications

The subject matter of this application is related to the subject matter of application serial number AA/AAA,AAA entitled, "System and method for Identifying the Workload Queued by a Monitor" filed on October 22, 1999 by Jean-Jaques Heller having the same assignee as this application and is incorporated herein by reference in its entirety.

Field of the Invention

The present invention is related to computer software and more specifically to Internet computer software.

Background of the Invention

Many sites on the World Wide Web have capacity to handle large numbers of simultaneous users. However, every web site has a finite capacity for a given level of

service. When this capacity is exceeded, the web site may operate below the level of service, or may stop operating at all.

If it is desirable to maintain the level of service of a web site, it may be desirable to delay or deny additional users access to the web site when the site is operating at or near capacity. Although the users who are delayed or denied access to the web site may be frustrated that they do not have access, the level of service is maintained for those who have access.

When a conventional site delays access to users, users are granted access to the site when capacity becomes available. Typically, capacity becomes available when a user using the site leaves the site. Shortly after a user leaves the site, another user who happens to attempt to gain entry at that moment is granted access to the site.

There are several problems with this approach. First, it does not operate fairly. Gaining access to such a web site is a chance occurrence. The users who have been waiting the longest do not stand a greater probability of gaining access to such a web site than other users who may gain access on their first attempt.

The second problem is related to the first one. Because users realize that the probability of gaining access to the web site is higher as the number of access attempts they make increases, users will repeatedly attempt to access the site. The repeated access attempts can increase the load on the web server, which may further reduce the capacity of the system, or even bring the system down.

Solutions to this problem have been elusive. One reason that solutions to this problem are elusive is caused by the architecture of conventional web sites. Conventional web sites are made scalable using a "server farm" containing multiple servers. Each server in the server farm handles a subset of the total number of users to the site. These servers operate somewhat independently of one another to build and format web pages and receive information from the users. Although each web server may access a central database to provide some of the information in the web pages they build, much of the work may be handled by each web server independently of the others. This architecture makes it difficult to place users in a conventional queue because a central database would be needed to maintain the queue. If the site can receive a large amount of traffic during peak periods, the

additional investment in equipment and software required to implement and manage the central queue can be substantial.

A queue is particularly inefficient for a web site because many users may abandon their attempt to access a web site if they have to wait for access. A place in a conventional queue must be maintained even for users who abandon their attempt at access, or the connection between the web server and the user must be maintained to allow for detection of an abandoned attempt. Either alternative can be an inefficient use of resources.

What is needed is a method and apparatus that can provide access to a web site in a fair manner that does not reward repeated attempts, does not require a queue and adjusts for any rate of abandonment of user attempts.

Summary of Invention

When a method and apparatus receives a request for a web page, a timestamp, designating the time the user made the request, is supplied with one or more commands to rerequest the page. The commands may be in the form of a program, applet or script that will cause the browser that made the original request to send periodic rerequests for the web page. The method and apparatus maintains a list of the number of waiting users who have received timestamps

and did not miss retrying at approximately a time the user was expected to retry. The method and apparatus periodically calculates or receives a capacity of additional users who may be allowed to access the web page. The method and apparatus periodically scans the list of waiting users until it identifies a "cutoff" timestamp for which the number of users issued a timestamp prior to the cutoff timestamp is approximately equal to the available capacity of the system. When a rerequest is made, the method and apparatus receives or retrieves the timestamp issued to the user in response to the user's original request. If the timestamp received with, or retrieved following, the rerequest is not later than the cutoff timestamp, the user is provided access to the web page. Browsers sending rerequests for which the timestamp is later than the cutoff timestamp will send additional rerequests until the timestamp is not later than the cutoff timestamp, which is continuously advanced forward as additional capacity becomes available.

Because the timestamp is used to gate access, no central queue need be maintained. Because access is granted based on the time the first request was made, repeated attempts to gain access to the web page do not

help a user access the web page any faster, reducing the chance of repeated attempts.

Brief Description of the Drawings

Figure **1** is a block schematic diagram of a conventional computer system.

Figure **2** is a block schematic diagram of a system for providing web pages according to one embodiment of the present invention.

Figure **3** is an illustration of an LSP status daemon file according to one embodiment of the present invention.

Figure **4** is a block schematic diagram of a web server according to one embodiment of the present invention.

Figure **5** is a block schematic diagram of an LSP status daemon according to one embodiment of the present invention.

Figure **6** is a flowchart illustrating a method of providing server status and a cutoff timestamp according to one embodiment of the present invention.

Figure **7** is a flowchart illustrating a method of delaying access to a user requesting service from one or more servers according to one embodiment of the present invention.

Figure **8** is a schematic diagram of a table stored in timestamp storage of Figure **5** according to one embodiment of the present invention.

Figure **9** is a flowchart illustrating a method of recording a maximum number of users waiting for access to a portion of a web site according to one embodiment of the present invention.

Figure **10** is a flowchart illustrating a method of calculating a number of users waiting for access to a portion of a web site according to one embodiment of the present invention.

Detailed Description of a Preferred Embodiment

The present invention may be implemented as computer software on a conventional computer system. Referring now to Figure **1**, a conventional computer system **150** for practicing the present invention is shown. Processor **160** retrieves and executes software instructions stored in storage **162** such as memory, which may be Random Access Memory (RAM) and may control other components to perform the present invention. Storage **162** may be used to store program instructions or data or both. Storage **164**, such as a computer disk drive or other nonvolatile storage, may provide storage of data or program instructions. In one

embodiment, storage **164** provides longer term storage of instructions and data, with storage **162** providing storage for data or instructions that may only be required for a shorter time than that of storage **164**. Input device **166** such as a computer keyboard or mouse or both allows user input to the system **150**. Output **168**, such as a display or printer, allows the system to provide information such as instructions, data or other information to the user of the system **150**. Storage input device **170** such as a conventional floppy disk drive or CD-ROM drive accepts via input **172** computer program products **174** such as a conventional floppy disk or CD-ROM or other nonvolatile storage media that may be used to transport computer instructions or data to the system **150**. Computer program product **174** has encoded thereon computer readable program code devices **176**, such as magnetic charges in the case of a floppy disk or optical encodings in the case of a CD-ROM which are encoded as program instructions, data or both to configure the computer system **150** to operate as described below.

In one embodiment, each computer system **150** is a conventional Sun Microsystems Ultra 1 Creator computer running the Solaris 2.5.1 operating system commercially

available from Sun Microsystems of Mountain View, California, although other systems may be used.

Referring now to Figure **2**, a system **200** for providing web pages is shown according to one embodiment of the present invention. Router **210** includes a conventional router and distribution software to distribute among web servers **220-224** requests received at input/output **208**, coupled to the Internet or an intranet, for access to one or more web pages. Web servers **220-224** are conventional web servers, with additions as noted below. Web servers **220-224** process requests from users, which may involve access to a central server **260**. The central server **260** may be a conventional mainframe computer system or a group of servers, any of which may include a conventional database.

Web servers **220-224** request and receive data used to build the web pages they serve from server **260** via server interfaces **230, 231**. In one embodiment, all the data needed by web servers **220-224** is on server **260** and in another embodiment, data may be obtained from several sources (not shown) in addition to server **260**. In such case, in one embodiment, web servers **220-224** are provided access to the data that does not reside on server **260** via server **260** which has connections to the other sources of

data. In another embodiment, web servers **220-224** access the data directly via connections to other servers (not shown). Server interfaces **230, 231** can convert requests received from web servers **220-224** into a format suitable for server **260** and provide the formatted request to the server **260**. Server interfaces **230, 231** receive the response from the server **260** and may convert the response into a format suitable for use by the web servers **220-224** and provide the formatted response to the web servers **220-224**. In one embodiment, server interfaces **230, 231** may perform buffering, queuing and other functions as well.

As shown in the Figure, there are five web servers **220-224**, two server interfaces **230, 231** and one server **260** although any number of these may be used according to the present invention. In another embodiment (not shown), there are 45 web servers and 9 server interfaces.

LSP status daemon **240** periodically tests each server interface **230, 231** to identify its status. In one embodiment, LSP status daemon **240** tests the status of each server interface **230, 231** by sending one or more requests for service and investigating any response received from that server interface **230, 231**. LSP status daemon **240** then identifies the status of the server interface **230, 231**. In

one embodiment, a "Y" status indicates the server interface **230** or **231** is running and available, a status of "N" indicates the server interface **230** or **231** is not available for reasons other than serving as an interface (e.g. it is not operating or running a maintenance procedure) and a status of "Q" indicates the server interface **230** or **231** is operational but busy performing server interface functions for a web server **220-224** as described above.

If the LSP status daemon **240** does not receive a response after a certain number of attempts such as three attempts, the responses are not received within a certain time, or the responses are error messages or messages indicating the server interface **230, 231** is busy processing other requests, the LSP status daemon **240** internally records the server interface **230** or **231** as having a status equal to "Q" indicating the server interface **230, 231** has no available capacity. If a response is received or received within an acceptable time, LSP status daemon **240** marks the server interface **230** or **231** as having a status equal to "Y" indicating the server interface **230, 231** has available capacity.

LSP status daemon **240** thus uses the status of each of the server interfaces **230, 231** as a proxy to identify

available capacity of the server **260** in the embodiment described above. However, in other embodiments, other identifiers of the status of the server interfaces **230, 231** such as performance/utilization statistics can be used, and in still another embodiment, the server **260** can be investigated directly using techniques similar to those described above for the server interfaces **230, 231** (e.g. performing a test that can fail to show the server **260** is busy or requesting performance/utilization statistics).

In one embodiment, the status in LSP status daemon **240** of a server interface **230** or **231** may be manually adjusted to a value of "Y" or "Q" or "N". Such a manual status adjustment may be made using the conventional keyboard and mouse of conventional workstation **234**. If the status in LSP status daemon **240** of a server interface **230, 231** is manually adjusted to "N", LSP status daemon **240** will not test that server interface **230, 231** as described above until the status for that server interface **230, 231** is manually adjusted back to "Y" or "Q".

LSP status daemon **240** periodically builds and maintains a file called the LSP status daemon file which provides an indication of the status of each server interface **230, 231** obtained as described above. Referring

momentarily to Figure **3**, an illustration of an LSP status daemon file **300** is shown according to one embodiment of the present invention. The file **300** contains an identifier of each server interface **230, 231** and an indication of the status ("Y", "Q" or "N") of the server interface **230, 231** as described above.

Referring again to Figure **2**, LSP status daemon **240** holds the LSP status daemon file in storage such as conventional memory or disk storage, and makes the file available to web servers **220-224**. In one embodiment, the file is stored in LSP status daemon **240**, and web servers **220-224** can read or request from LSP status daemon **240** the file or portions of the file for use as described below. In another embodiment, LSP status daemon **240** periodically provides to web servers **220-224** copies of some or all of the LSP status daemon file for use as described below.

Referring now to Figures **2** and **4**, an example one **220** of the web servers **220-224** is shown according to one embodiment of the present invention. Input/output **402** is coupled to LSP status daemon **240** and receives the LSP status daemon file and other information as described below. The LSP status daemon file is received and stored in LSP status daemon file storage **410** in the embodiment in

which the file is provided to and stored on the web servers **220-224**. The LSP status daemon **240** also provides at input/output **402** a cutoff timestamp, generated and used as described below. The cutoff timestamp is received and stored by cutoff timestamp storage **412**, which may include conventional memory or disk storage.

Queue requirement generator **420** reads the cutoff timestamp from cutoff timestamp storage **412** and determines if queuing is necessary. In one embodiment, queue requirement identifier **420** reads the cutoff timestamp when requested by user request router **414** as described below.

In one embodiment, if the cutoff timestamp has a special value, such as a negative value, or a value of 00:01, queue requirement identifier **420** will indicate that no queuing is required. Otherwise, queue requirement identifier **420** will indicate that queuing is required as described below.

In another embodiment, queue requirement identifier **420** reads the LSP status daemon file in LSP status daemon file storage **410** to determine if queuing is required. In one embodiment, queuing is required if the ratio of server interfaces with a "Y" status to server interfaces with "Y" or "Q" status is less than a threshold, such as 50 percent.

Users wishing to access the capabilities of the system **200** access the system **200** via input/output **404** coupled to user request router **414** and router **210**. When a user provides a request for a page not requiring a log in or other information from server **260**, the page is served to the user by user request router **414**.

As used herein, a log in page is used as a gate to other pages that may require the server interfaces **230, 231**. Until the user gains access to the log in page, other types of access to the server interfaces **230, 231** may be limited or denied. However, other types of pages may be used as the gate, or no gate may be used at all as described below.

If the user requests at input/output **404** a page requiring a log in, user request router **414** requests the queue requirement from queue requirement identifier **420**. Queue requirement identifier **420** determines and indicates whether queuing is required as described above. If queuing is not required, user request router **414** routes the user to log in/run manager **424**.

If queuing is required, user request router **414** will determine whether a timestamp is available from the user. A timestamp is available if the user had made an earlier

request and that request was denied and the user did not abandon the request as described below. The timestamp, described in more detail below, identifies the time the user first tried to access the log in/run manager **424** but was denied. There are many ways of determining whether such a timestamp is available. In one embodiment, the user's timestamp is stored as a cookie on his computer system as described in more detail below. User request router **414** sends a conventional command to retrieve the cookie (e.g. @cookies=split in perl, or document.cookie.split in Javascript or another similar command to retrieve a cookie) to request the cookie containing the timestamp. In another embodiment, the timestamp, if available as described below, is provided by the user's browser with the request that requires the use of the log in/run manager **424** via conventional CGI techniques.

If a timestamp is not available, the user is attempting to access log in/run manager **424** on a basis other than retrying after a delay as described below. User request router **414** signals cookie/applet generator **422**. Cookie/applet generator **422** requests via input/output **408** coupled to operating system (not shown) the time or the time and date from the system clock. Cookie/applet

generator **422** receives the time or the date and time at input **408**. In one embodiment, cookie/applet generator **422** generates a cookie containing the system time or the date and time and stores it on the user's system via user request router **414** and input/output **404** using conventional CGI or Javascript scripting techniques or other similar techniques. Cookie/applet generator **422** also generates a program to be run on the user's system that will resend the request requiring the log in/run manager **424**. The program may cause the request to be resent only once or periodically until the request is successful or the user abandons the request until the request is successful or the user abandons the request. The period which the user's system waits to resend the request is referred to as the "retry period". The program may be a Java applet, Javascript script or other similar program that is capable of operation by a browser or otherwise.

In another embodiment, the timestamp is not sent separately to the user as a cookie. Instead, the timestamp is embedded in the applet or script and so no cookie is provided by cookie/applet generator **422**. Cookie/applet generator **422** sends the applet, and optionally the cookie, to user request router **414**, which sends the applet and optionally the cookie to the user as a web page or a web

page and a cookie via input/output **404**. The user's browser running the script or applet returns the timestamp and optionally, the retry period it received, to user request router **414** with all subsequent requests it makes until the applet or script is halted by the user or the request is successful as described below.

In one embodiment, cookie/applet generator **422** sends a retry period that is static. All cookies and/or applets it generates have the same retry period, such as 30 seconds. In another embodiment, cookie/applet generator **422** receives at input **423** the retry period it should send from retry rate calculator **530**, described below, via input/output **402**.

The applet or script will cause the user's conventional browser such as the conventional Internet Explorer Browser commercially available from Microsoft Corporation of Redmond, Washington, running on a conventional computer system (not shown) such as a conventional Pentium compatible computer system available from Dell Computer Corporation of Round Rock, Texas, to delay for the retry period and then generate another request to the user request router **414** that will require the log in/run manager **424** unless the user stops the operation of the script or applet making additional

requests. The applet or script sends the timestamp and optionally the retry period, with this request as noted above. The applet or script may also perform other useful functions, such as displaying to the user that he or she has been queued and will get served as capacity is available.

If the timestamp is available, either from the request or from a cookie, user request router **414** sends the timestamp to timestamp compare **430**. Timestamp compare **430** compares the timestamp it receives from user request router **414** with the timestamp in cutoff timestamp storage **412** to determine if the timestamp received from user request router **414** is earlier than the timestamp in cutoff timestamp storage **412**. If it is earlier, timestamp compare **430** signals user request router **414**, which sends the user's request to log in/run manager **424**, and may send an indication via input/output **404** to the script or applet retrying that the retry attempt has succeeded. If the timestamp is not earlier, timestamp compare **430** signals user request router **414** that this is the case. User request router **414** provides the timestamp received to cookie applet generator **422**. Cookie/applet generator **422** generates another applet with the original timestamp in one embodiment, or in another embodiment sends an indication

back to the user leaving the original applet on the user's system to retry using the timestamp. The same or a different retry period may also be provided by cookie applet generator **422** allowing the retry period to change for each retry. For example, if the retry period is calculated according to how close the timestamp is to the cutoff timestamp as described below, the user's browser may receive a different retry period than was used before.

If a different retry period will be sent, cookie/applet generator **422** requests and receives a new retry period from LSP status daemon **240** via input/output **402** as described below. In one embodiment, the timestamp for which the retry period is being requested is provided to LSP status daemon **240** with the request for the new retry period.

In one embodiment, cookie/applet generator **422** maintains the number of retries that have been attempted. This is performed by sending as part of the cookie or applet the number of retries attempted and receiving this number from the user with each retry. The number is originally set to 0 when the first attempt to log in is performed. Each time the number of retries is received or retrieved as part of the cookie by cookie/applet generator

422, the number of retries is incremented and returned to the user to be provided with the next retry.

Each time the timestamp is sent to a user, cookie/ applet generator **422** sends via input/output **402**to stroke counter **532** for use as described below the time stamp it receives or, in the case in which no timestamp was received, the time stamp it generates.

When a user's request is sent to log in manager **424**, log in manager **424** handles the log in procedure in conjunction with user request router **414**. User request router **414** is coupled via input/output **406** to server interfaces **230**, **231** of Figure **2** to provide requests to, and retrieve information from, server **260** of Figure **2** as described above. In one embodiment, there may be multiple log in/run managers for each web server as shown in Figure **4**.

Referring now to Figures **2** and **5**, an LSP status daemon **240** is shown according to one embodiment of the present invention. Among the functions performed by LSP status daemon **240** is the building of the LSP status daemon file, the calculation of the cutoff timestamp and calculation of retry periods if the retry periods are variable. Each of these will now be described.

Server interface monitor **510** maintains a list of each server interface **230, 231** in the system **200**. To create the LSP status daemon file, server interface monitor **510** monitors the server interfaces **230, 231** by periodically sending via input/output **504** coupled to each server interface **230, 231** a request to perform work. If the request causes server interface **230** or **231** to return an error via input/output **504**, or if the request is not performed or is not performed within a specific period after the request is sent, server interface monitor **510** marks that server interface as busy as described above. In one embodiment, three consecutive failed (e.g. error message is returned, not performed or not performed within a specific time) attempts are made before a server interface **230** or **231** is marked as busy. The server interface **230** or **231** is marked as busy by inserting a "Q" associated with that server in the list of server interfaces **230, 231** server interface monitor **510** maintains. If the request is answered at input/output **504**, server interface monitor **510** inserts a "Y" associated with that server interface **230, 231** in its list to indicate the server is available.

If server interface monitor **510** receives at input **506**, which is coupled to a conventional input device such as a

keyboard and mouse, an indication that server interface monitor **510** should consider a specified server interface to be up, busy or down, server interface monitor **510** provides a status of "Y", "Q", or "N", respectively associated with the specified server in the list it maintains.

Periodically, such as every 30 seconds, when server interface monitor **510** updates its list, it signals LSP status daemon file builder **512**.

LSP status daemon file builder **512** retrieves the list from server interface monitor **510** along with the status of "Y", "Q" or "N" for each server interface **230, 231** when signaled by server interface monitor **510**. LSP status daemon file builder **512** builds the LSP status daemon file described above using the list it retrieves and stores it into LSP status daemon file storage **514** which may be conventional memory or disk storage. When it is finished storing the file, LSP status daemon file builder **512** signals LSP status daemon file provider **516**, cutoff timestamp calculator **524** and capacity calculator **522**.

In one embodiment, LSP status daemon file provider **516** provides to each web server **220-224** via input/output **502**, coupled to input output **402** of Figure **4** the LSP status daemon file in LSP status daemon file storage **514** when it

is signaled by LSP status file builder **512**. In another embodiment, LSP status daemon file storage **514** is connected to input/output **502** to allow each web server **220-224** to retrieve the LSP status daemon file directly. In such embodiment, LSP status daemon file provider **516** need not be used.

Stroke counter **530** receives the timestamps provided as described above by cookie/applet generator **422** of Figure **4**. In one embodiment, cookie/applet generator **422** provides, and stroke counter **530** receives and stores into timestamp storage **520**, both the timestamp and the retry period for each timestamp issued. If a retry is received but not serviced by log in manager **424** as described above, cookie/applet generator **422** will again provide the timestamp and optionally the retry period, to timestamp storage **520** each time a retry is attempted.

Stroke counter **532** maintains in timestamp storage **520** a table of 30 second periods for the day and the number of timestamps that were issued in the thirty second period and are potentially waiting to retry. Each time a timestamp is received by stroke counter **532**, it updates the table.

Referring momentarily to Figure **8**, a table is shown according to one embodiment of the present invention.

Column **810** holds one 30 second period for each row. Although only two rows are shown, there may be one row for each 30 second period in each day and the table may span multiple days or a single day and be reused. Columns **812-820** refer to retry 0, the issuance of the initial timestamp, and columns **830-838** refer to the first retry.

Column **812** holds the number of users who were issued timestamps during the period of column **810**. In the sample table, 100 users were issued timestamps during period 1, from 0:00 to 0:30, and 90 users were issued timestamps during period 2, following 0:30 to 1:00. Column **814** holds the retry period most recently issued to the user and the cumulative total of all retry periods. In the table, a one minute retry period was issued with the initial timestamp, and the cumulative total of all retry periods is 0. Column **816** holds the number of users who abandoned since their last try (which is always 0 in the case of column **816** because no last try exists). Column **818** holds the number of users who have expressly abandoned since the last retry by clicking an abort button that may be part of the applet generated by cookie/applet generator **422** again 0 for both periods. Column **820** contains the system time the timestamp for the first user was received by stroke counter **532**.

Column **830** holds the number of users who retried during their first retry. In the table, 50% of the users have retried during both periods. Column **832** holds the last retry period and the cumulative retry period, both one minute for this example in both 30 second periods. Column **834** holds the number or percentage of users who did not retry. In the table, column **834** is shown expressed as a number. Column **836** records the number of express abandon requests since the last retry (in such case, user request router **414** receives and passes the abort request to cookie/applet generator **422** which provides the timestamp and the number of the retry to stroke counter **532** for recordation in the proper column of the table in timestamp storage **520**. The proper column will be one plus the number of the retry, e.g. column **836** for a number of retry equal to 0, etc.). In the table, 20% of the users issued timestamps expressly aborted from the time they were issued the timestamp to the time immediately prior to their first retry. Column **838** holds the time during which the first timestamp for the period corresponding to the first retry was received.

As shown in the table, space for recording statistics for one retry exists, but in other embodiments, statistics for an unlimited number of retries may be recorded in the

table for each 30 second period. In another embodiment, the two sets of columns (one set per retry) operate as a circular buffer, with each set used for alternate retries and only the most recent initial try or retry stored in the table. For example, statistics for the second retry would be stored in columns **812-820**.

In one embodiment, stroke counter **532** outputs via output **540** some or all of the statistics in the table to allow for statistical analysis.

Referring now to Figure **9**, a method of recording a maximum number of users waiting for access to a portion of a web site is shown according to one embodiment of the present invention. A timestamp, and optionally a retry period and number of retries is received **910**.

If the timestamp received in step **910** corresponds to an express abandonment request, the number of express abandons is updated **914** in the proper position of the table as described above and the method continues at step **910**. If the timestamp is the first in the 30 second period for the number of the retry **914**, the retry period and cumulative amount of all retry periods from the prior retry is added and stored **918** as the new cumulative amount of all retry periods for the current retry, along with the retry

period received in step **910**. Otherwise, the method continues at step **920**. The number of retries is updated and stored **920** in the proper column of the table according to the retry number received in step **910**, and the method continues at step **910**.

Referring again to Figure **9**, thus at any given time, the latest column of each row in the table in timestamp storage **520** contains the maximum number of people who may still be waiting for service and have a timestamp in the 30 second period for the row. The reason that it takes slightly longer than the exact retry period to receive the timestamps for a retry is to account for delays either due to download speeds or other factors.

It is not necessary to receive the retry number with each request, as the retry number can be deduced based on the cumulative retry period and the last retry period of the prior set of columns in the table. However, because users with timestamps in the first half of the 30 second period may attempt retry N before users with timestamps in the second half of the 30 second period attempt retry N-1, stroke counter will also look at where each timestamp is located along the 30 second period in order to determine which set of columns in the table to update.

In one embodiment, capacity calculator **522** determines the available capacity of the system using the LSP status daemon file stored in LSP status daemon file storage **514**. Capacity calculator **522** determines the available capacity based on the number or percentage of server interfaces **230, 231** that are listed in the LSP status daemon file as being available or using other information in the file. For example, each server interface **230, 231** may be determined to have available capacity of 200 users if the server interface has a 'Y' status, and if both server interfaces are listed with a 'Y' status in the LSP status daemon file, capacity calculator identifies the available capacity of the system as 400 users. Capacity calculator **522** calculates the number of additional users for which the system has available capacity and provides this number to cutoff timestamp calculator **524**.

In one embodiment, capacity calculator **522** adjusts its calculations of available capacity by watching the results of operation of the apparatus of the present invention. If a certain percentage of server interfaces **230, 231** continue to have 'Y' status, capacity calculator will increase the estimated available capacity of each 'Y' status server interface **230, 231**. If another percentage of server interfaces **230, 231** have 'N' status, capacity calculator

will decrease the estimated available capacity of each 'Y' status server interface **230, 231.**

In another embodiment, capacity calculator **522** receives an indication of the system capacity at input **508,** and performs any necessary calculations to convert the indication into a number of users. The indication may be generated using the apparatus described in the copending application serial number AA/AAA,AAA or using any other method. In still another embodiment, capacity calculator **522** obtains the raw data that is used to generate the LSP status daemon file and uses that data instead of the file to determine the number of available server interfaces **230, 231.**

Figure **10** illustrates a method of calculating a number of users waiting for access to a portion of a web site according to one embodiment of the present invention. Referring now to Figures **5** and **10,** cutoff timestamp calculator **524** periodically retrieves **1010** from operating system (not shown) the current system clock time, and identifies the available capacity of the system as described above. Cutoff timestamp calculator **524** selects **1012** the 30 second period immediately following the period corresponding to the system clock as the start of the table

of timestamps in timestamp storage **520** in one embodiment (or another time sufficiently in the past so that it would be unlikely that users were still retrying with timestamps issued before that time), or another starting point (such as the time of or near the cutoff timestamp) in another embodiment.

Cutoff timestamp calculator **524** accumulates the number of users corresponding to each thirty second period in the table from the beginning of the table of timestamps in timestamp storage **520** until it has accumulated a number of users is greater than the capacity of the system received by capacity calculator **522**. The number of users is retrieved from the column in the table based on the current system time and the expected retry time for each 30 second period. In some cases, the number of users is in the right most set of columns for some 30 second periods, in other cases it is the set of columns immediately prior to the right most set of columns and in still other cases, the number is 0. Cutoff timestamp calculator **524** retrieves the proper number from the table as follows.

For each 30 second period, starting with the latest set of columns for the 30 second period, if the system clock is less than **1014** a few seconds past the system clock

recorded in the set of columns plus the retry period recorded for that set of columns, the number of users in the set of columns preceding the latest set of columns is retrieved **1016** from the table. If the system clock is more **1018** than the system clock recorded in the set of columns plus the retry period plus a little over 30 seconds, none of the users with timestamps for the 30 second period have retried, and the number of users waiting to retry is 0 **1020**. Otherwise **1018**, the number of users recorded in the latest set of columns in the table is retrieved from the table **1022**. For example, in the table in Figure **8**, at 2:02, the number of users in the first 30 second period is 50, and the number of users in the second 30 second period is 90. At 5:00, if the table in Figure 8 is unchanged, the number of users in both 30 second periods is 0.

The number of users is accumulated to a total number of users and the total is compared with the system capacity. If the total number of users is less than the system capacity **1018**, the next chronological 30 second period is selected **1028** and the method continues at step **1012**. Otherwise cutoff timestamp calculator **524** selects **1030** the prior period and retrieves from timestamp storage **520** the time corresponding to the end of the thirty second period immediately preceding the thirty second period in

which the number of users exceeds the capacity of the system.

In another embodiment, instead of using the end of the prior period, the cutoff timestamp is calculated by subtracting from the ending time of the 30 second period in which the total exceeds the available capacity an amount of time equal to 30 seconds multiplied by the excess of the total above the system capacity divided by the number of users in that 30 second period. Thus, step **1040** replaces step **1030** in the figure, as indicated by the dashed lines. Other forms of interpolation may also be used.

Cutoff timestamp calculator **532** provides **1032** the cutoff timestamp at output **502** to each web server **220-224**. In one embodiment, step **1032** is performed only if the cutoff timestamp changes in value. The method continues at step **1010**.

Because the number of users in each row of the table in timestamp storage **520** is based on a snapshot in time that may be inaccurate an instant later due some of those users abandoning, the number of users represents the maximum number of users who may retry. To obtain a more accurate estimate of the number of users who actually will retry, it may be desirable to adjust the number of users

based on the rate at which users are abandoning in order to get a more accurate and use the estimate in place of the number of users in the table when calculating the cutoff timestamp. In one embodiment, instead of accumulating the number of users in the table as described above, cutoff timestamp calculator **524** accumulates an adjusted **1024** number of users recorded in each 30 second period in the table based on some or all of: 1) the amount of time between the last time the number of users was recorded or updated for the period and the time of the system clock retrieved as described above, 2) the amount of time between the end of the 30 second period and the time of the system clock, and 3) the abandon rate for the users in that period (or an overall abandon rate for all users). For example, if 50% of the users have abandoned in the 180 seconds between the end of the period and the most recent recordation or update of the number of users still retrying, and cutoff timestamp calculator **524** is calculating the cutoff timestamp 18 seconds after the most recent recordation or update, cutoff timestamp calculator **524** will accumulate only (1-18/180) or .9 times the number of users in the table for that period. This adjustment assumes that users are abandoning at a steady rate. Cutoff timestamp calculator **524** may use other functions such as

exponential functions to calculate the adjustment. In this manner, cutoff timestamp calculator **524** can extrapolate a more accurate number of users waiting to retry in each thirty second period.

It is also accurate to assume that additional users will abandon between the time of the system clock and the time the user's browser actually retries, and cutoff timestamp calculator **524** may make such a further adjustment based on the expected time of the retries that are pending in each thirty second period. Using the last example, if the average retry for a 30 second period will occur in another 18 seconds, cutoff timestamp calculator **524** may further adjust the number of users in that period by another ten percent.

In one embodiment, after step **1028**, if cutoff timestamp calculator **524** finds the list of timestamps in timestamp storage **520** empty **1034**, cutoff timestamp calculator **524** indicates **1036** that queuing is not required as described above and the method continues at step **1010**.

Referring now to Figure **5**, retry period calculator **530** is used to calculate different retry periods based on the number of users waiting to retry in the embodiment in which the retry rate is variable.

In one embodiment, retry period calculator **530** provides a retry period based on the difference between the cutoff timestamp and the timestamp for which the retry period is being issued. In such embodiment, cutoff timestamp calculator **524** provides the cutoff timestamp to retry period calculator **530** when it provides it at input/output **502** as described above. Retry period calculator **530** receives from cookie/applet generator **422** the timestamp of the user for which the retry period is being calculated.

In one embodiment, the difference between the timestamp being issued and the cutoff timestamp is rounded down to the nearest second and used as the retry period by retry period calculator **530**. In another embodiment, a fraction, such as 0.5 or 0.75 of the difference is used as the retry period. In another embodiment, the difference is computed using one of the techniques described above, but then adjusted based on the trend of the difference between the cutoff timestamp and the system clock, which cutoff timestamp generator **530** retrieves from an operating system, not shown via a connection not shown. In such embodiment, retry period calculator **530** stores one or more of the past values of the difference between the cutoff timestamp and the system clock and identifies a trend. The retry period

is adjusted higher than the difference between the timestamp of the user for whom the cutoff timestamp is being issued and the cutoff timestamp if the trend is increasing and adjusted lower than such difference if the trend is decreasing. If the trend is unchanged or indeterminiant, the retry period is not adjusted in this fashion.

Because the retry period is calculated based on the timestamp of the user for whom the retry period is being calculated, users closer to the cutoff timestamp are issued a retry period that is shorter than those being issued a timestamp initially.

In one embodiment, retry period calculator **530** only calculates a single retry period for all timestamps that will be stored in the same 30 second period in timestamp storage **520**.

In one embodiment, the prior retry period calculated is stored by retry period calculator **530** in case a retry arrives in the first few seconds after the most recent retry period is calculated and it has a timestamp that is near the end of the 30 second period. In such case, the prior retry period may be used by retry period calculator **530** to respond to the request that includes such timestamp

on the theory that the request has arrived late due to the delays described above. In another embodiment, once the retry period is calculated for a 30 second interval, all requests with timestamps in that interval use the new interval.

Referring now to Figure **6**, a method of providing server status and a cutoff timestamp is shown according to one embodiment of the present invention. One or more servers are monitored **610** as described above to determine their availability. This may be accomplished by sending one or more commands and waiting for acknowledgement of the commands sent, or waiting for the result of the command. The results of the monitoring are used to determine **612** the server status as described above. The server status is provided **614**, for example in the form of the LSP status daemon file or in any other form.

The available system capacity is received or determined **616**. The determination of the capacity of the system may be performed as described above, such as is described in copending application serial number AA/AAA,AAA or using any other method. A cutoff timestamp is calculated or identified, and then provided **618** as

described above. The beginning timestamp may be marked **620** as described above and the current time may be recorded.

A determination is made **622** as to whether a sufficient time period has elapsed since the last time the server status or the cutoff timestamp was provided, or any other of the steps of Figure **6** were performed. The determination may include comparing a system clock against a stored time of performance of such step of Figure **6**, such storage having been performed as part of that step. If a sufficient period of time has elapsed **622**, the method continues at step **610**. Otherwise, the method waits **624** and step **622** is performed again.

The method of figure **6** illustrates a sequence between two groups of steps: **610-614** and **616-620**. In another embodiment, these two groups are performed in the reverse order (steps **616-620**, then steps **610-614**). In another embodiment, each group is performed independently of the other, with steps similar to step **622-624** performed after performance of each group, with the methods repeating at the first step of each group. In yet another embodiment, after one or both groups are performed, the method repeats from the first step of each group, and no determination or

wait similar to steps **622-624** are performed following either or both groups.

Referring now to Figure **7**, a method of delaying access to a user requesting service from one or more servers is shown according to one embodiment of the present invention. A request is received **710**. The request is identified to determine whether the request requires a log on **712**. If the request does not require a log on, **712**, the request is processed **714** and the method continues at step **710**. Otherwise, the method continues at step **716**.

One or more status indications of the server or servers are received **716** as described above. The status indication can be in the form of the LSP status daemon file or any other form that describes the status of one or more servers. If the status indications received in step **716** indicate that queueing is not required **718** as described above, the request is processed **720**, for example by sending it to the server or a server interface as described above, and the method continues at step **710**. Otherwise, the method continues at step **722**.

If a timestamp was not received **722** with the request, an applet, with or without a cookie, which may include a timestamp and a statically or variably calculated retry

period, is generated and provided **728** to the user sending the request received in step **710** as described above. The timestamp and optionally the retry period is stored as part of step **730** for use in identifying the cutoff timestamp as described above.

If a timestamp was received with the request **722**, a cutoff timestamp is received **724**, having been generated as described above. If the timestamp received with the request is not later than, or not later than or equal to the cutoff timestamp **726**, the method continues at step **720**. Otherwise, the method continues at step **728** in one embodiment.

In another embodiment, in place of continuing at step **728**, a rejection is transmitted or no response to the request is transmitted or another form of response is transmitted to the device that sent the request that will cause the device to wait for another retry period and send the request again. In one such embodiment, the retry period may be updated in the response, so that the device which sent the request will wait for a different retry period than was used for the request received in step **710**. The method then continues at step **710**.

In one embodiment, if a log on request is processed, all subsequent requests from that user are processed without performing steps **718** and **720** and in another embodiment, subsequent requests also use steps **718** and **720**.

What is claimed is:

1. A method of processing a first request for web page, comprising:

receiving the first request for the web page; and

transmitting at least one command to send a second request for the web page, and a first timestamp.

2. The method of claim 1 wherein the transmitting step is responsive to an existence of a second timestamp received with the request.

3. The method of claim 2 comprising the additional steps of:

identifying a third timestamp; and

responsive to the second timestamp received with the request, processing the request for the web page responsive to the second timestamp and the third timestamp.

4. The method of claim 3 wherein the identifying the third timestamp step is responsive to a capacity of at least one selected from at least one server and a device coupled to the at least one server.

5. The method of claim 4 additionally comprising incrementing at least one of a plurality of counters responsive to the first request.

6. The method of claim 5 wherein each of the plurality of counters corresponds to a range of time different from the other plurality of counters.

7 The method of claim 6 wherein the identifying the third timestamp step is additionally responsive to at least one of the plurality of counters.

8. The method of claim 5 comprising the additional steps of:

receiving a notification of abandonment of at least one selected from the first request and the second request; and

decrementing at least one of the plurality of counters.

9. The method of claim 3 wherein the identifying the third timestamp step comprises sending a command to at least one selected from at least one server and a device coupled to the at least one server.

10. The method of claim 3 wherein the identifying the third timestamp step comprises building a file comprising a status of at least one selected from at least one server and at least one device coupled to the at least one server.

11. The method of claim 1, wherein the transmitting step is responsive to a type of the first request.

12. The method of claim 1, additionally comprising transmitting computer readable program code devices configured to cause a computer to send the second request responsive to the indicator transmitted.

13. The method of claim 1 wherein the computer readable program code devices configured to cause the computer to send the second request responsive to the indicator transmitted comprise at least one selected from a javascript script and a java applet.

14. A computer program product comprising a computer useable medium having computer readable program code embodied therein for processing a first request for web page, the computer program product comprising:

computer readable program code devices configured to cause a computer to receive the first request for the web page; and

computer readable program code devices configured to cause a computer to transmit at least one command to send a second request for the web page, and a first timestamp.

15. The computer program product of claim 14 wherein the computer readable program code devices configured to

cause a computer to transmit are responsive to an existence of a second timestamp received with the request.

16. The computer program product of claim 15 additionally comprising computer readable program code devices configured to cause a computer to:

identify a third timestamp; and

responsive to the second timestamp received with the request, process the request for the web page responsive to the third timestamp and the second timestamp.

17. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp are responsive to a capacity of at least one selected from at least one server and a device coupled to the at least one server.

18. The computer program product of claim 17 additionally comprising computer readable program code devices configured to cause a computer to increment at least one of a plurality of counters responsive to the first request.

19. The computer program product of claim 18 wherein each of the plurality of counters corresponds to a range of time different from the other plurality of counters.

20 The computer program product of claim 19 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp are additionally responsive to at least one of the plurality of counters.

21. The computer program product of claim 18 additionally comprising:

computer readable program code devices configured to cause a computer to receive a notification of abandonment of at least one selected from the first request and the second request; and

computer readable program code devices configured to cause a computer to decrement at least one of the plurality of counters.

22. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp comprise sending a command to at least one selected from at least one server and a device coupled to the at least one server.

23. The computer program product of claim 16 wherein the computer readable program code devices configured to cause a computer to identify the third timestamp comprise computer readable program code devices configured to cause

a computer to build a file comprising a status of at least one selected from at least one server and at least one device coupled to the at least one server.

24. The computer program product of claim 14, wherein the computer readable program code devices configured to cause a computer to transmit are responsive to a type of the first request.

25. The computer program product of claim 14, additionally comprising computer readable program code devices configured to cause a first computer to transmit computer readable program code devices configured to cause second computer to send the second request responsive to the indicator transmitted.

26. The computer program product of claim 14 wherein the computer readable program code devices configured to cause the computer to send the second request responsive to the indicator transmitted comprise at least one selected from a javascript script and a java applet.

27. An apparatus for processing a first request for a web page, the apparatus comprising:

a user request router having an input coupled to an apparatus input operatively coupled for receiving the first request, the user request router for providing at an output

a signal responsive to the first request received at the user request router input; and

a cookie/applet generator having an input coupled to the user request router output for receiving the signal, the cookie/applet generator for providing a first output coupled to an apparatus output a first indicator of at least one time to send a second request for a web page.

28. The apparatus of claim 27, wherein the first request comprises a second indicator of time, and the user request router provides the signal at the user request router output responsive to the second indicator of time.

29. The apparatus of claim 28, wherein the cookie/applet generator provides at a second output a third indicator of time corresponding to the first indicator of time, the apparatus additionally comprising:

a timestamp storage for having an input coupled to the cookie/applet generator third output for receiving the third indicator of time, the timestamp storage for storing the third indicator of time and a set of fourth indicators of time and for providing the third indicator of time and the set of fourth indicators of time at an input/output; and

a cutoff timestamp calculator having an input operatively coupled for receiving an indicator of capacity, the cutoff timestamp calculator for selecting and providing at an output a timestamp from the set of fourth indicators of time responsive to the capacity; and

wherein the user request router additionally comprises a cutoff timestamp input coupled to the cutoff timestamp calculator output and the user request router provides the signal additionally responsive to the timestamp received at the cutoff timestamp input.

30. The apparatus of claim 27, wherein the cookie/applet generator additionally provides at the cookie/applet generator first output computer readable program code devices configured to cause a computer to send the second request responsive to the indicator.

31. The apparatus of claim 30 wherein the computer readable program code devices configured to cause the computer to send the second request responsive to the indicator transmitted comprise at least one selected from a Javascript script and a Java applet.

1. A method of determining a number of users in a first group waiting for access to a portion of a web site, the method comprising:

receiving a request for access to the portion of the web site from each of a plurality of a second group of users;

providing during a first period a first number of at least one first identifier to each of the plurality of users in the second group to indicate that access to the portion of the web site is denied;

receiving during a second period following the first period a second number of at least one second identifier corresponding to the first identifier from each of the users in the first group; and

determining the number responsive to the second number of at least one second identifier received.

2. The method of claim 1 additionally comprising the steps of:

receiving during the second period at least one second request from at least one user not in the first group for access to the portion of the web site; and

providing at least one third identifier to the at least user not in the first group.

3. The method of claim 2 wherein the at least one first identifier is provided responsive to a time of each of the at least one request.

4. The method of claim 1 wherein the determining step is responsive to the at least one second identifier received following the end of the second period.

5. The method of claim 4 wherein the determining step is responsive to at least one second after the end of the second period.

6. The method of claim 4 wherein the determining step is additionally responsive to the first number prior to the end of the second period.

7. The method of claim 1:

additionally comprising the step of identifying one selected from a third number and a rate of users abandoning the request for access; and

wherein the determining step is responsive to at least one selected from the third number and the rate.

8. A computer program product of determining a number of users in a first group waiting for access to a portion of a web site, the computer program product comprising:

computer readable program code devices configured to cause the computer to receive a request for access to the portion of the web site from each of a plurality of a second group of users;

computer readable program code devices configured to cause the computer to provide during a first period a first number of at least one first identifier to each of the plurality of users in the second group to indicate that access to the portion of the web site is denied;

computer readable program code devices configured to cause the computer to receive during a second period following the first period a second number of at least one second identifier corresponding to the first identifier from each of the users in the first group; and

computer readable program code devices configured to cause the computer to determine the number responsive to the second number of at least one second identifier received.

9. The computer program product of claim 8 additionally comprising:

computer readable program code devices configured to cause the computer to receive during the second period at least one second request from at least one user not in the first group for access to the portion of the web site; and

computer readable program code devices configured to cause the computer to provide at least one third identifier to the at least user not in the first group.

10. The computer program product of claim 9 wherein the computer readable program code devices configured to cause the computer to provide the at least one first identifier are responsive to a time of each of the at least one request.

11. The computer program product of claim 8 wherein the computer readable program code devices configured to cause the computer to determine are responsive to the at least one second identifier received following the end of the second period.

12. The computer program product of claim 11 wherein the computer readable program code devices configured to cause the computer to determine are responsive to at least one second after the end of the second period.

13. The computer program product of claim 11 wherein the computer readable program code devices configured to

cause the computer to determine are additionally responsive to the first number prior to the end of the second period.

14. The computer program product of claim 8:

additionally comprising computer readable program code devices configured to cause the computer to identify one selected from a third number and a rate of users abandoning the request for access; and

wherein the computer readable program code devices configured to cause the computer to determine are responsive to at least one selected from the third number and the rate.

Abstract of the Disclosure

A method and apparatus queues users accessing a service via the world wide web without maintaining a queue. When a user requests the service, if the service is busy, a web server does not allow access to the service: instead it configures the user's web browser via a script or applet to periodically retry requesting the service. A timestamp designating a time the user requested the service or other similar time is provided to the user and a count of timestamps issued in different periods are maintained by the web server. The web server periodically determines a cutoff timestamp corresponding to an available capacity of the apparatus using the count of timestamps it maintains. When the user retries requesting the service, the browser provides to the web server the timestamp it received from the server. If the timestamp provided is less than or equal to the cutoff timestamp, the user is allowed access to the service, otherwise, the users browser is instructed to retry at a specified interval.